Java 程序设计中的数据交换和通信技术

杨少波 （中国科学院计算技术研究所 100080）

摘要：本文通过实例介绍实现 Java 语言程序设计中的数据交换和通信技术的几种方法，并给出具体实现的关键部分程序代码。

关键词：Java 程序 数据交换 程序通信

本文通过实例并借助于 Visual J + + 1.1 版开发平台，介绍实现 Java 语言程序设计中的数据交换和通信技术的几种方法，并给出具体实现的关键部分程序代码，从而达到增强程序的交互功能。

1. 网页中各个 Java Applet 小程序相互通信

Java Applet 是内嵌到网页文档中的小程序，利用它可以实现动态交互性能较佳的网页[1]，但如果网页中同时内嵌若干个 Applet 小程序，它们之间又如何实现。通信，同时某一个 Applet 小程序怎样与其网页文档所属的浏览器进行通信等技术问题？本文论述这方面的具体技术实现。

(1)Java API 中的 AppletContext 接口介绍。通过使用 java.applet.AppletContext 接口中的成员方法，用户可以获取网页中内嵌的 Applet 小程序的环境信息，调用该接口中的成员方法 getApplet(String name)可以查找给定名称的 Applet 小程序是否存在并将它返回；而使用成员方法 getApplets()则可以获取网页文档中所有内嵌的 Applet 小程序(它返回一个 Enumeration 接口对象，然后调用其中的成员方法 nextElement()可以轮询各个内嵌的 Java Applet 小程序)。

(2)命名网页文档中各个内嵌的 Java Applet 小程序。Java Applet 在缺省时可以不用命名它，但为实现各个 Applet 小程序之间进行相互通信，在将它们内嵌到网页文档时应加以命名。在 HTML 网页文档中主要可以采用如下两种方式：其一是在＜Applet＞＜/Applet＞标签中指定其 Name 属性；另一手段则是通过＜Param＞标签来指定一个 Name 参数，并将其值设为指定的 Applet 小程序名称字符串。如下文描述代码所示：

```
＜applet code = JavaDemo.class codebase = c:/javademo alt = "Java Applet A"
align = center width = 320 height = 240 ＞
＜Param name = "name" value = "JavaDemo" ＞ ＜/applet＞
```

本文采用第一种形式内嵌各个 Applet 小程序。描述代码为：

```
＜html＞＜head ＞＜title＞This is Yang's HomePage ＜/title＞
＜/head＞＜body＞
...其他的 HTML 网页文档语句省略
＜applet code = JavaDemo.class codebase = c:/javademo alt = "Java Applet A".
name = JavaDemo align = center width = 320 height = 240 ＞
＜param name = Receive value = "JavaApplet"＞
＜param name = AppletString value = "朋友，您好！这儿是 JAVA Applet A."＞.
＜/applet＞
＜applet code = JavaApplet.class codebase = c:/javademo alt = "Java Applet B"
name = JavaApplet align = middle width = 260 height = 240＞
＜/applet＞＜/body＞＜/html＞
```

(3)通信机制及实现的程序代码

一个 Java Applet 可以向同一网页文档中内嵌的其他 Applet 小程序发送消息，即首先通过调用 java.applet.AppletContext 接口中的成员方法 getApplet()搜索网页文档中指定名称的 Applet 小程序是否存在；利用返回的 Applet 类的对象，调用指定 Applet 小程序中的 public 成员方法(本文为 callBackFunction())，并通过成员方法的形参和返回值实现数据交换和通信。关键部分的程序代码如下：

```
import java.applet.*;
import java.awt.*;
public class JavaDemo extends Applet implements Runnable
```

```
} // ... 其他的成员方法程序代码省略
  public boolean action(Event event, Object obj) //行为事件响应程序代码
  {
    String displayString = getParameter("AppletString");
    if(event.target = = stopDialog.IDCSTOP) //指定行为事件发生时
    {
      Applet receiveApplet =
        getAppletContext().getApplet(getParameter("Receive"));
//调用指定名称的 JavaApplet 程序中的 public 成员方法,并传送字符串参数
    ((JavaApplet) receiveApplet).callBackFunction(displayString);
    }
    // ... 其他行为事件响应程序代码
    return false;
    }
}
```

2. Java Applet 小程序与网页浏览器相互通信

在 Internet 网应用中不仅要能实现各个内嵌的 Applet 小程序之间相互通信,有时也需要与其浏览器进行通信,以提高交互功能。

(1)实现通信的有关 Java API 类及成员方法。网页中的 Java Applet 可以利用 java.applet.Applet 类的成员方法 getDocumentBase()返回网页文档的统一资源定位器 URL 地址,调用 getCodeBase()返回 Applet 小程序本身的 URL 地址,从而获取与网页文档有关的来源信息;showStatus(String msg)则可以在浏览器的状态条窗口中显示指定提示文字串,据此可使用户获知当前的操作执行状态。

在 Applet 小程序中调用 getParameter(String name)可以获知网页文档中由 <Param> 标签内的 Name 属性所指定的 Value 值;当然浏览器也可通过调用 Applet 小程序中的各个成员方法如 init()、start()、stop()等程序代码,驱动内嵌的 Applet 小程序完成指定的任务。

(2)驱动浏览器以显示指定 URL 地址的网页文档。利用 java.applet.AppletContext 接口中的两个重载形式的成员方法 showDocument(URLurl),可以替换当前正在浏览的主页文档为指定 URL 地址的主页文档;showDocument(URL url, String target)则可以在指定显示位置处显示出指定 URL 地址的主页文档。

```
import java.applet.*;
import java.awt.*;
import java.net.*;
public class JavaApplet extends Applet implements Runnable
{ //..其他的成员方法程序代码省略
  public void callBackFunction(String senderString)
  {
    receiveEdit.setText(senderString);
  //显示另一个 Applet 传送来的字符串,让 IE 浏览器显示指定的 HTML 文档
    getAppletContext().showDocument(getDocumentBase());
    if(getCodeBase().getHost() = = null)
      showStatus("Host is Null String");
    showStatus(getCodeBase().getHost()); //在状态条中显示主机名
  }
}
```

3. Java Application 应用程序之间相互通信及数据交换

目前由于 Java API 类库不及 Visual C++ 类库丰富,Java 普通应用程序之间相互通信交换数据还无法借助象 Visual C++ Windows 应用程序设计所采用的剪贴板、动态数据交换 DDE、对象嵌入与链接 OLE2 等技术手段,但 Java SDK API 提供有 java.io 包,其中包含有对管道流的支持类 PipedInputStream 和 PipedOutputStream。本文论述利用管道流来实现两个 Java 普通应用程序之间数据交换和通信。

(1)管道流。利用管道 IO 流可以将一个线程的输出数据输送到另一个线程的输入端,直接通过内存进行数据交换,从而避免采用临时磁盘文件方式,提高了数据交换速度和效率。

管道输入流是一个通信管道的接收端,由 PipedInputStream 类来描述;而管道输出流则是指一个通信管道的发送端,在 Java SDK API 中由 PipedOutputStream 类描述;其通信机制如下:一个线程通过管道输出流 PipedOutputStream 类中的成员函数 write(byte [], int off, int len)发送数据,而输入端线程则通过管道接口由 PipedInputStream 管道输入流类中的成员函数 read(byte [], int off, int len) 读取数据,并且双方应保持同步协调。

(2)联接管道的两个端口。Java SDK API 提供两种方式实现管道的两个端口之间的相互联接,联接成功后

便可进行数据交换,本文采用第二种方式。

方法一:通过管道流类的构造函数

直接将某一个线程的输出作为另一个线程的输入,只需要在定义管道流对象时将一个对象作为另一个类的构造函数的参数。

```
PipedInputStream pipInput = new PipedInputStream();
PipedOutputStream pipOutput = new PipedOutputStream(pipInput);
```

方法二:利用管道流类中的成员方法 connect()

调用管道 IO 流类中成员方法 connect(),并将对方管道流类的一个对象作为其参数,达到相互联接。

```
PipedInputStream pipInput = new PipedInputStream();
PipedOutputStream pipOutput = new PipedOutputStream();
pipOutput.connect(pipInput);
```

(3)数据交换及实现的程序代码。本文在一个 Java 应用程序中创建两个线程,一个线程向管道输出端口输出数据,而另一个线程则根据需要从管道输入端读取数据,关键部分的程序代码如下:

```
import java.io.*;
public class DataExchangeTest
  { public static void main(String args[]
    { try
      { PipedInputStream pipInput = new PipedInputStyream();
        PipedOutputStream pipOutput = new PipedOutputStream();
        pipOutput.connect(pipInput); //联接管道的两个端口
//创建管道的两个端口线程并启动该两个线程
        OutputDataThread outputThread = new OutputDataThread(pipOutput);
        outputThread.start();
        InputDataThread inputThread = new InputDataThread(pipInput);
        inputThread.start();
      }
      catch(IOException e) //捕获 IO 流异常
      {
      System.out.println("Now Pipe Input Output Error" + e);
      }
      }
    }
class OutputDataThread extends Thread //输出线程
{ PipedOutputStream outPip;
  byte buffer[] = new byte[1024];
  int offset, byteNum;
  OutputDataThread(PipedOutputStream outPip)
  {
      this.outPip = outPip;
  }
  // ...其他功能性函数及程序代码
  public void run()
  {
  try
    {
  // ...其他程序代码,在线程体中通过管道的输出端口输出数据
    outPip.write(buffer, offset, byteNum);
  // ...其他程序代码
    outPip.close();
    }
  catch(IOException e)
    {
      System.out.println("Output Data Thread Error" + e);
    }
  }
}
class InputDataThread extends Thread //输入线程
{
  PipedInputStream inPip;
  byte buffer[] = new byte[1024];
  int offset, byteNum;
  InputDataThread(PipedInputStream inPip)
  {
    this.inPip = inPip;
  }
  // ...其他功能性函数及程序代码
  public void run()
  {
```

```
try
  {
// 在线程体中通过管道的输入端口读取数据,然后使用数据
  while(inPip.read(buffer,offset,byteNum)! = -1)
  {
// ...其他程序代码(使用数据 )
  }
  inPip.close();
  }
catch(IOException e)
  {
    System.out.println("Input Data Thread Error" +
e);
  }
}
```

4. 结束语

除可采用本文介绍的各种方法进行相互通信及数据交换之外,还可使用 java.net 包中的 API 类及成员方法(如 Socket 类、URL 类和 URLConnection 等类)通过网络进行通信。

参考文献

[1] 杨少波,Java Applet 调用 ActiveX 控件的技术,计算机系统应用,1998 年第 5 期

[2] 王克宏,刘波 等编著,Java 语言 API 类库,1997 年 5 月,北京:清华大学出版社

(来稿时间:1998 年 6 月)